What does it take to certify conversion?

Meven Lennon-Bertrand

FSCD 2025

University of Cambridge
Department of Computer Science and Technology

What am I trying to do?

Verifying the verifier

We keep telling the world they should verify their critical code...

We keep telling the world they should verify their critical code...
A lot of the verification ecosystem relies on proof assistant kernels...

We keep telling the world they should verify their critical code...
A lot of the verification ecosystem relies on proof assistant kernels...

Yet we still don't have verified kernels!*

*For dependent types: CANDLE exists and it's really cool

We keep telling the world they should verify their critical code…
A lot of the verification ecosystem relies on proof assistant kernels…

Yet we still don't have verified kernels!*

The programs are (relatively) simple…

Verifying the verifier

We keep telling the world they should verify their critical code…
A lot of the verification ecosystem relies on proof assistant kernels…

Yet we still don't have verified kernels!*

The programs are (relatively) simple…

But the reasons why they work are very complicated!

Where I'm coming from

- MetaRocq: Rocq in Rocq
- *Martin-Löf à la Coq*: a place to experiment

Where I'm coming from

- MetaRocq: Rocq in Rocq
- *Martin-Löf à la Coq*: a place to experiment

Key characteristics:

- We care about the **actual implementation**
- We need to manipulate **extrinsically typed terms**
- We are **fundamentally limited** by Gödel's 2nd incompleteness theorem

Logical relations

Irrelevant computational content

Meta-theory

Models

Algebraic/ Presentation-free/ Intrinsically typed

Execution/Extraction

Verification

Invariants

Datastructures

Logical relations

Irrelevant
computational content

Execution / Extraction

?

Meta-theory

Verification

Models

Invariants

Algebraic/
Presentation-free/
Intrinsically typed

Datastructures

The algorithms (and their specification)

Specification

Rules for each term/type former
$(\Pi, \Sigma, \mathrm{Id}, \mathcal{U}, \mathbb{N}, \bot ...)$ +

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T \cong T'}{\Gamma \vdash t : T'}$$

Specification

Rules for each term/type former
$(\Pi, \Sigma, \mathrm{Id}, \mathcal{U}, \mathbb{N}, \bot\ldots)$ +

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T \cong T'}{\Gamma \vdash t : T'}$$

Algorithm

$$\xcancel{\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T \cong T'}{\Gamma \vdash t : T'}}$$

Bidirectional: rules for each term former integrate (some) conversion.

Conversion

Declarative specification

Arbitrarily mixing:

- Refl./Sym./Trans.
- Congruences
- Computation (β)
- Extensionality (η)

Typed!

Declarative specification

Arbitrarily mixing:

- Refl./Sym./Trans.
- Congruences
- Computation (β)
- Extensionality (η)

Typed!

Type-directed algo.

Alternate

1. β-reduction to whnf
2. Type-directed η
3. Head congruences

Declarative specification

Arbitrarily mixing:

- Refl./Sym./Trans.
- Congruences
- Computation (β)
- Extensionality (η)

Typed!

Type-directed algo.

Alternate

1. β-reduction to whnf
2. Type-directed η
3. Head congruences

"Untyped" algo.

Alternate

1. β-reduction to whnf
2. Term-directed η
3. Head congruences

Conversion

Declarative specification

Arbitrarily mixing:

- Refl./Sym./Trans.
- Congruences
- Computation (β)
- Extensionality (η)

Typed!

Type-directed algo.

Alternate

1. β-reduction to whnf
2. Type-directed η
3. Head congruences

+ closer to specification
+ supports fancier rules
- slower

"Untyped" algo.

Alternate

1. β-reduction to whnf
2. Term-directed η
3. Head congruences

+ faster
+ simpler (?)
- further from spec.

The plan

What's in a decision procedure?

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

What's in a decision procedure?

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$

What's in a decision procedure?

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$
1. soundness: $p\ d = \mathrm{true} \Rightarrow P\ d$
2. completeness: $P\ d \Rightarrow p\ d = \mathrm{true}$
3. profit!

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$
1. soundness: $p\ d = \mathrm{true} \Rightarrow P\ d$
2. completeness: $P\ d \Rightarrow p\ d = \mathrm{true}$
3. profit?

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \text{true}) \lor (p\ d = \text{false})$

   How do you know that the type-checker terminates?

1. soundness: $p\ d = \text{true} \Rightarrow P\ d$

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. profit?

What's in a decision procedure?

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \text{true}) \vee (p\ d = \text{false})$

   How do you know that the type-checker terminates?

1. soundness: $p\ d = \text{true} \Rightarrow P\ d$

   Look at the trace of the type-checker

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. profit?

What's in a decision procedure?

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$

   How do you know that the type-checker terminates?

1. soundness: $p\ d = \mathrm{true} \Rightarrow P\ d$

   Look at the trace of the type-checker

2. completeness: $P\ d \Rightarrow p\ d = \mathrm{true}$

   reflexivity $\simeq$ normalisation

3. profit?

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

1. positive soundness: $p\ d = \text{true} \Rightarrow P\ d$

   Look at the trace of the type-checker

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \rightharpoonup \mathbb{B}$$

1. positive soundness: $p\ d = \mathrm{true} \Rightarrow P\ d$
   Look at the trace of the type-checker
2. negative soundness: $p\ d = \mathrm{false} \Rightarrow \neg(P\ d)$
   Look (harder) at the trace of the type-checker

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \to \mathbb{B}$$

1. **positive soundness**: $p\ d = \mathrm{true} \Rightarrow P\ d$
   Look at the trace of the type-checker
2. **negative soundness**: $p\ d = \mathrm{false} \Rightarrow \neg(P\ d)$
   Look (harder) at the trace of the type-checker
3. **termination**: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$
   Still hard, of course...

Better abstract nonsense

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \mathrm{true} \Rightarrow P\ d$
   Look at the trace of the type-checker
2. **negative soundness**: $p\ d = \mathrm{false} \Rightarrow \neg(P\ d)$
   Look (harder) at the trace of the type-checker
3. **termination**: $(p\ d = \mathrm{true}) \vee (p\ d = \mathrm{false})$
   Still hard, of course...

Now we have a plan

What meta-theory do we need?

The good properties

Injectivity and no-confusion of type constructors

If $\Gamma \vdash T \cong T'$ and T, T' are weak-head normal form, then:

- $T = \mathbb{N} = T'$
- or $T = \Pi\, x{:}\, A.B$, $T' = \Pi\, x{:}\, A'.B'$, with $\Gamma \vdash A' \cong A$ and $\Gamma, x{:}\, A' \vdash B \cong B'$
- or ...
- or T, T' are both neutral, and $\Gamma \vdash T \cong T' : \mathcal{U}$

Any non-diagonal case is impossible (*no-confusion*).

Injectivity and no-confusion of type constructors

Injectivity and no-confusion at $\mathbb{N}$

If $\Gamma \vdash n \cong n' : \mathbb{N}$ and n, n' are weak-head normal forms, then:

- $n = 0 = n'$
- or $n = \mathrm{S}(t)$, $n' = \mathrm{S}(t')$, with $\Gamma \vdash t \cong t' : \mathbb{N}$
- or n, n' are both neutral.

The good properties

Injectivity and no-confusion of type constructors

Injectivity and no-confusion at $\mathbb{N}$

Injectivity and no-confusion at $\mathcal{U}$

...

Injectivity and no-confusion of type constructors

Injectivity and no-confusion at $\mathbb{N}$

Injectivity and no-confusion at $\mathcal{U}$

Injectivity of neutral eliminators*

If $\Gamma \vdash n \cong n' : T$ and n and n' are neutrals, then

- $n = x = n'$
- or $n = m\ u$, $n' = m'\ u'$ with $m \cong m'$ and $u \cong u'$
- or $n = \mathrm{rec}_{\mathbb{N}}(m, x.P, t_0, x.y.t_S)$, $n' = \mathrm{rec}_{\mathbb{N}}(m', x.P', t_0', x.y.t_S')$, and ...

* See paper/talk to me for subtleties.

Injectivity and no-confusion of type constructors

Injectivity and no-confusion at $\mathbb{N}$

Injectivity and no-confusion at $\mathcal{U}$

Injectivity of neutral eliminators*

Deep normalisation

Every well-typed term is deeply normalising at its type.
Every well-formed type is deeply normalising.

	Positive soundness	Negative soundness (typed conversion)	Negative soundness (untyped conversion)	Termination
Injectivity of type constructors	×	×	×	×
Term-level injectivities		×*	×*	
Normalisation				×

* Not *quite* the same for neutrals

	Positive soundness	Negative soundness (typed conversion)	Negative soundness (untyped conversion)	Termination
Injectivity of type constructors	×	×	×	×
Term-level injectivities		×*	×*	
Normalisation				×

Rocq (Ap)Proved

	Positive soundness	Negative soundness (typed conversion)	Negative soundness (untyped conversion)	Termination
Injectivity of type constructors	×	×	×	×
Term-level injectivities		×*	×*	
Normalisation				×

Rocq (Ap)Proved

Injectivities are the important properties

	Positive soundness	Negative soundness (typed conversion)	Negative soundness (untyped conversion)	Termination
Injectivity of type constructors	×	×	×	×
Term-level injectivities		×*	×*	
Normalisation				×

Injectivities are the important properties

Claim/conjecture: this analysis scales to realistic proof assistant kernels

How to prove the properties?

	Logical relation	Rewriting/Confluence	Gluing/Nf Model	Domain model
	[AÖV17; Adj+24]	[Tak95]/MetaRocq	[Ste21; BKS23]	[CH18]
Syntax	Raw	Raw	Intrinsic	Raw (Intrinsic?)
Weak ambiant theory	×	✓	×	✓
Normalisation	✓	×	✓	×
Scaling	×	✓	?	?
η laws	✓	×	✓	✓
	Most explored	Insane scaling	Formalisation currently difficult	Very unexplored

How to prove the properties?

	Logical relation [AÖV17; Adj+24]	**Rewriting/Confluence** [Tak95]/MetaRocq	**Gluing/Nf Model** [Ste21; BKS23]	**Domain model** [CH18]
Syntax	Raw	Raw	Intrinsic	Raw (Intrinsic?)
Weak ambiant theory	×	✓	×	✓
Normalisation	✓			×
Scaling	×			?
η laws	✓	×	✓	✓
	Most explored	Insane scaling	Formalisation currently difficult	Very unexplored

There is space for exploration!

How to prove the properties?

	Logical relation [AÖV17; Adj+24]	**Rewriting/Confluence** [Tak95]/MetaRocq	**Gluing/Nf Model** [Ste21; BKS23]	**Domain model** [CH18]
Syntax	Raw	Raw	Intrinsic	Raw (Intrinsic?)
Weak ambiant theory	×	✓	×	✓
Normalisation	✓			×
Scaling	×			?
η laws	✓			✓
	Most explored	Insane scaling	Formalisation currently difficult	Very unexplored

There is space for exploration!

… but I have peculiar requirements.

- You can (should!) separate meta-theory and implementation
- Injectivity properties are key, more so that normalisation
- You cannot beat Gödel, but you can salvage a lot with negative soundness
- There is space for new proof/formalisation techniques for meta-theory

You can (should!) **separate meta-theory and implementation**

Injectivity properties are key, more so that normalisation

You cannot beat Gödel, but you can salvage a lot with **negative soundness**

There is **space for new proof/formalisation techniques**

THANK YOU!

Bibliography

[AÖV17] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. **"Decidability of Conversion for Type Theory in Type Theory"**. In: *Proc. ACM Program. Lang.* POPL (Dec. 2017). DOI: 10.1145/3158111.

[Adj+24] Arthur Adjedj et al. **"Martin-Löf à la Coq"**. In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 2024. DOI: 10.1145/3636501.3636951.

[Tak95] M. Takahashi. **"Parallel Reductions in λ-Calculus"**. In: *Information and Computation* 1 (1995). DOI: 10.1006/inco.1995.1057. URL: https://www.sciencedirect.com/science/article/pii/S0890540185710577.

[Soz+24] Matthieu Sozeau et al. **"Correct and Complete Type Checking and Certified Erasure for Coq, in Coq"**. In: *Journal of the ACM* (Nov. 2024). DOI: 10.1145/3706056.

[Ste21] Jonathan Sterling. **"First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory"**. PhD thesis. Carnegie Mellon University, Nov. 2021. DOI: 10.5281/zenodo.6990769.

[BKS23] Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. **"For the Metatheory of Type Theory, Internal Sconing Is Enough"**. In: *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023*. 2023. DOI: 10.4230/LIPICS.FSCD.2023.18.

[CH18] Thierry Coquand and Simon Huber. **"An Adequacy Theorem for Dependent Type Theory"**. In: *Theory of Computing Systems* 4 (July 2018). DOI: